ЭЛЕМЕНТЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ

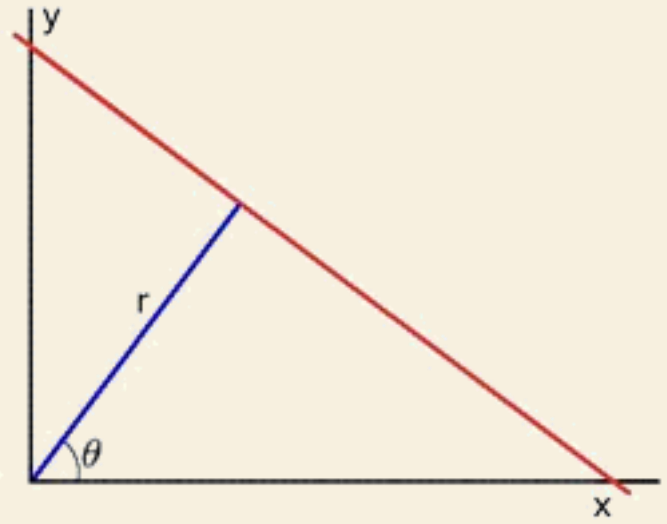
ПРЕОБРАЗОВАНИЕ ХАФА. РАЗНОЕ

ПРЕОБРАЗОВАНИЕ ХАФА

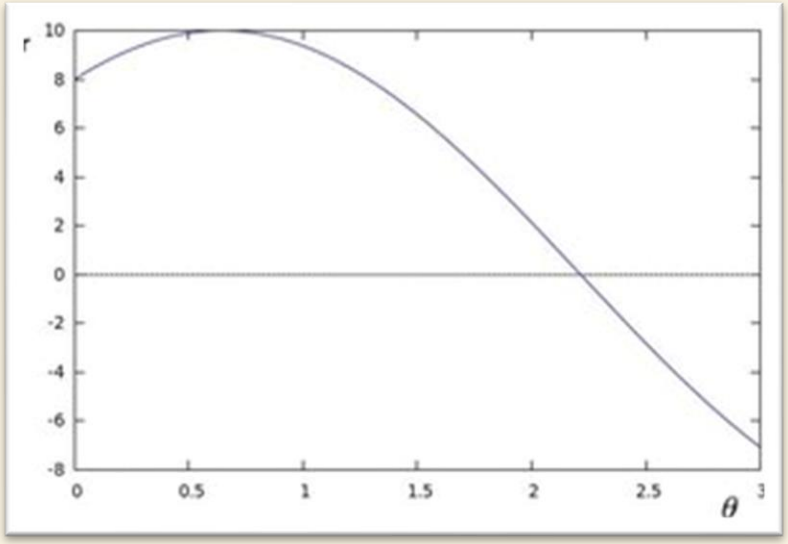
- Поиск прямых и окружностей на изображении
- Идея – построение аккумуляторов
- Мы переходим к пространству параметров и ищем в нём наилучшую оценку параметров
- Для этого сначала нам понадобится эту параметризация ввести
- Для прямой $X \cdot \cos\theta + Y \cdot \sin\theta = \rho$

ПРЕОБРАЗОВАНИЕ ХАФА

В КООРДИНАТНОМ ПРОСТРАНСТВЕ

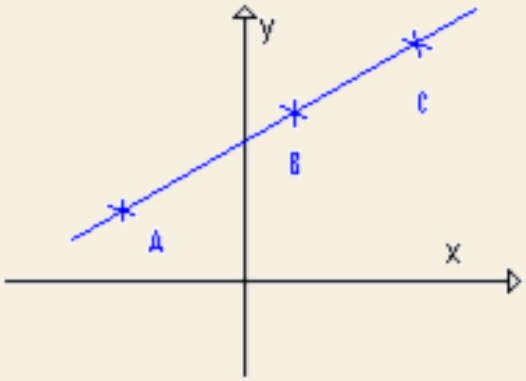


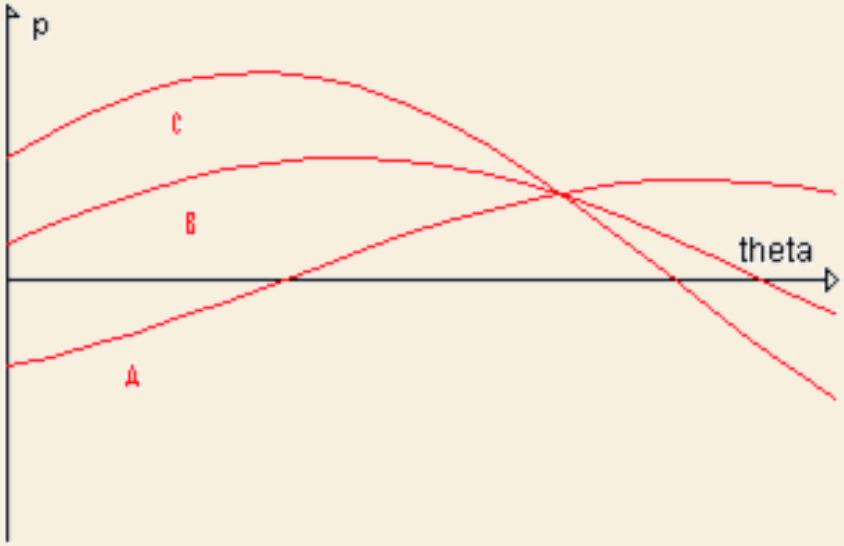
В ПРОСТРАНСТВЕ ПАРАМЕТРОВ



ПРЕОБРАЗОВАНИЕ ХАФА

Параметризация

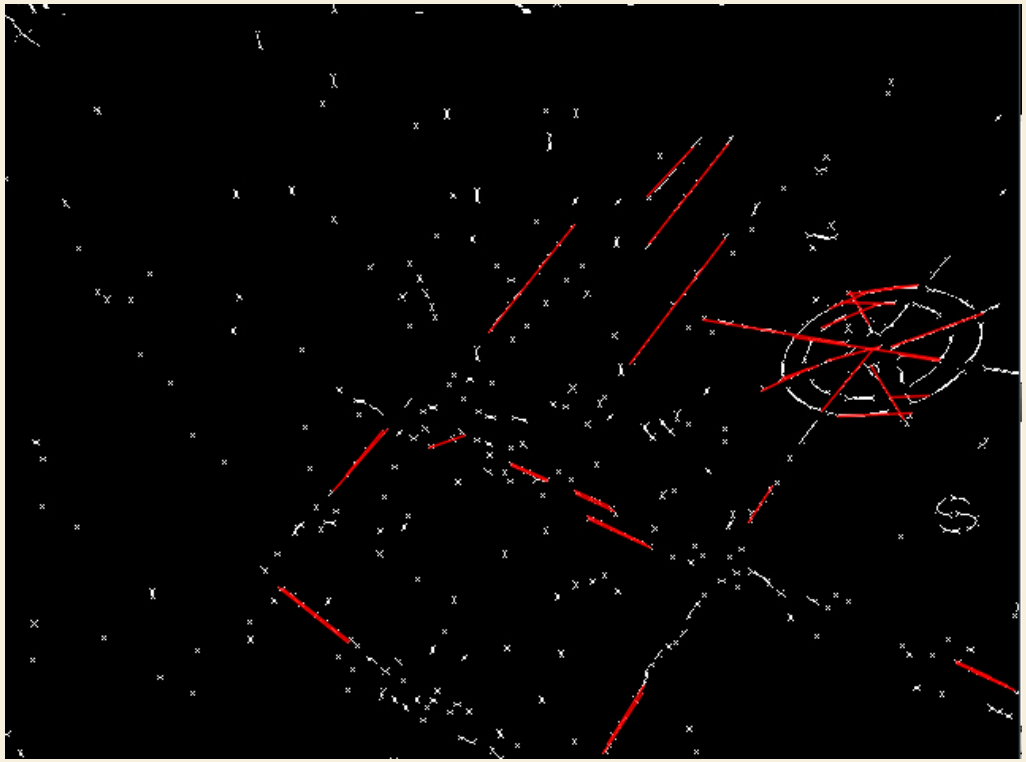




ПРЕОБРАЗОВАНИЕ ХАФА

```
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    Mat dst, cdst;

    string filename = “Picture1.png";
    Mat src = imread( filename , IMREAD_GRAYSCALE );
    if(src.empty()){
        return -1;
    }

    Canny(src, dst, 50, 200, 3);

    cvtColor(dst, cdst, COLOR_GRAY2BGR);
    cdstP = cdst.clone();

    vector<Vec2f> lines; // линии
    HoughLines(dst, lines, 1, CV_PI/180, 150, 0, 0 ); // детектор
    // рисование
    for( size_t i = 0; i < lines.size(); i++ )
    {
        float rho = lines[i][0], theta = lines[i][1];
        Point pt1, pt2;
        double a = cos(theta), b = sin(theta);
        double x0 = a*rho, y0 = b*rho;
        pt1.x = cvRound(x0 + 1000*(-b));
        pt1.y = cvRound(y0 + 1000*(a));
        pt2.x = cvRound(x0 - 1000*(-b));
        pt2.y = cvRound(y0 - 1000*(a));
        line( cdst, pt1, pt2, Scalar(0,0,255), 3, LINE_AA);
    }

    waitKey();
    return 0;
}
```

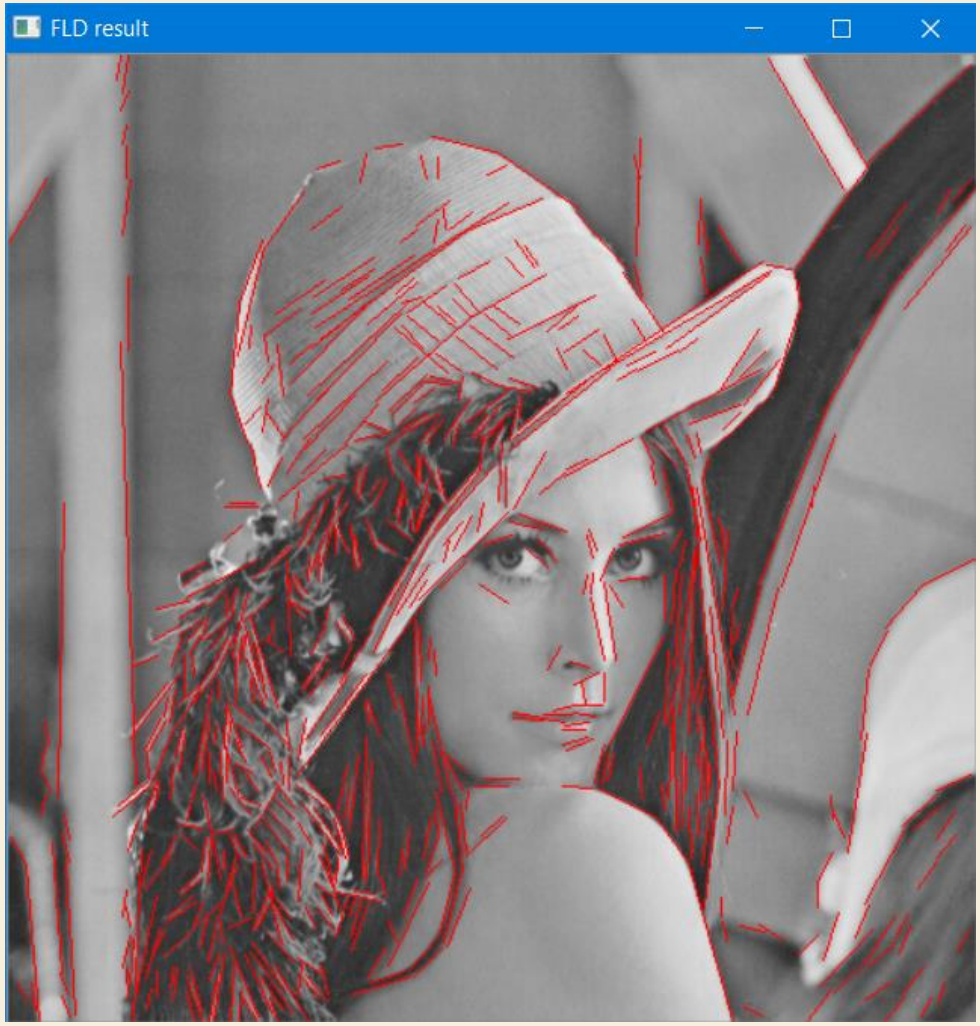
ПРЕОБРАЗОВАНИЕ ХАФА



```cpp
#include <iostream>

#include "opencv2/imgproc.hpp"
#include "opencv2/ximgproc.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"

using namespace std;
using namespace cv;
using namespace cv::ximgproc;

int main(int argc, char** argv)
{
    string filename = "Lenna.png";
    Mat image = imread(filename, IMREAD_GRAYSCALE);

    if( image.empty() )
    {
        return -1;
    }

    // length_threshold                    - минимальная длина
    // distance_threshold                  - минимальное расстояние
    // canny_th1                           - Нижний порог Кэнни
    // canny_th2                           - верхний порог Кэнни
    // canny_aperture_size                 - Апертура для оператора Собеля
    // do_merge                            - Объединяем ли мы сегменты
    int length_threshold = 10;
    float distance_threshold = 1.41421356f;
    double canny_th1 = 30.0;
    double canny_th2 = 90.0;
    int canny_aperture_size = 3;
    bool do_merge = true;
    Ptr<FastLineDetector> fld = createFastLineDetector(length_threshold,
            distance_threshold, canny_th1, canny_th2, canny_aperture_size,
            do_merge);
    vector<Vec4f> lines_fld;

    fld->detect(image, lines_fld);

    Mat line_image_fld(image);
    fld->drawSegments(line_image_fld, lines_fld);
    imshow("FLD result", line_image_fld);

    waitKey();
    return 0;
}
```

FAST LINE DETECTOR

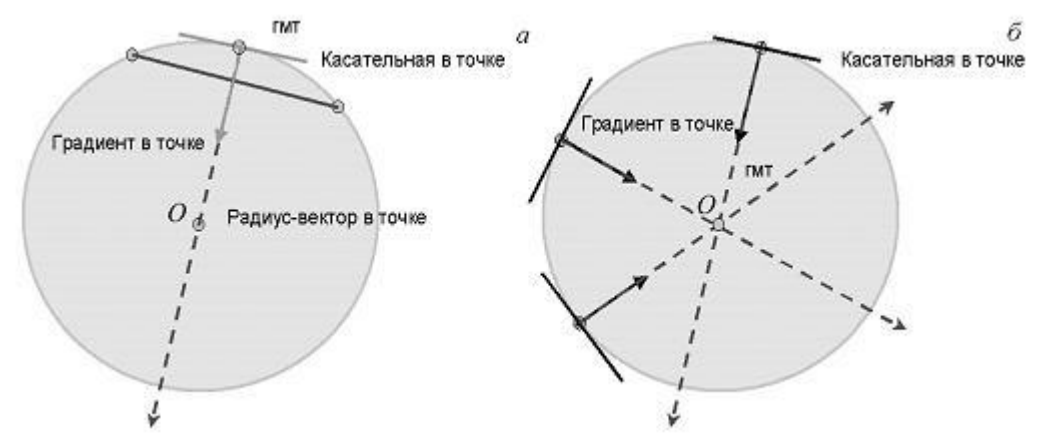
В дополнительных модулях



РЕЗУЛЬТАТ

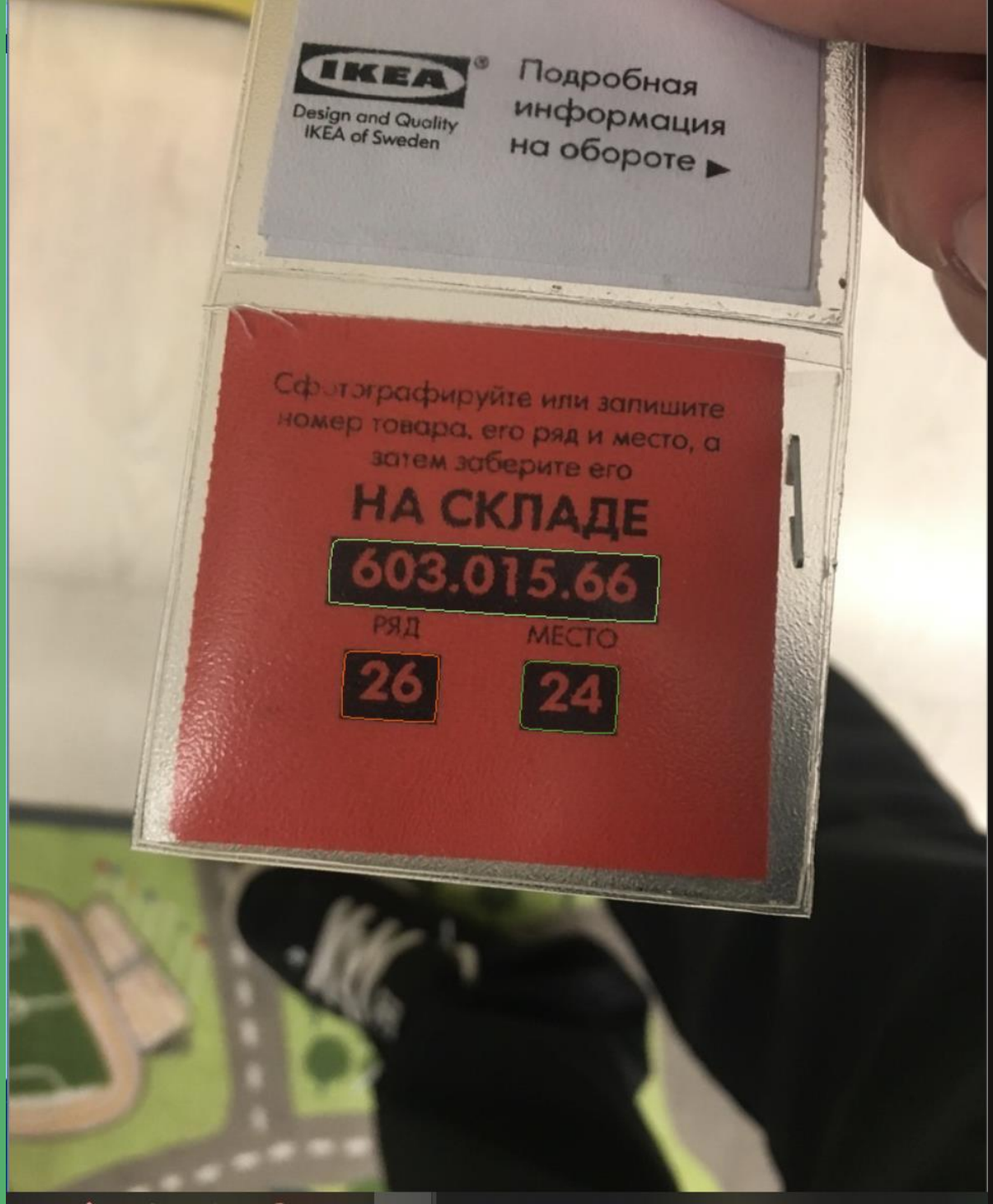
ПРЕОБРАЗОВАНИЕ ХАФА

- Аналогично – для окружностей
- Обычно предварительно находят края
- Параметризация $(x - x_0)^2 + (y - y_0)^2 = R^2$



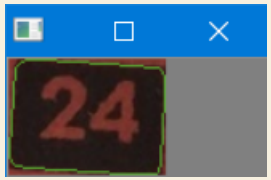
РЕАЛИЗАЦИЯ

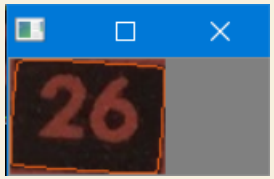
```
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    string filename =  "smarties.png";

    // Loads an image
    Mat src = imread(  filename , IMREAD_COLOR );

    if(src.empty()){
        return EXIT_FAILURE;
    }

    Mat gray;
    cvtColor(src, gray, COLOR_BGR2GRAY);

    medianBlur(gray, gray, 5);

    vector<Vec3f> circles;
    HoughCircles(gray, circles, HOUGH_GRADIENT, 1,
                 gray.rows/16, 100, 30, 1, 30 );

    for( size_t i = 0; i < circles.size(); i++ )
    {
        Vec3i c = circles[i];
        Point center = Point(c[0], c[1]);
        // circle center
        circle( src, center, 1, Scalar(0,100,100), 3, LINE_AA);
        // circle outline
        int radius = c[2];
        circle( src, center, radius, Scalar(255,0,255), 3, LINE_AA);
    }

    imshow("detected circles", src);
    waitKey();

    return EXIT_SUCCESS;
}
```



ПЯТАЯ ЗАДАЧА

Выделение областей





САМИ ОБЛАСТИ

Вот то, сего хотелось бы получить